

# Unification of XML Documents with Concurrent Markup

## Poster Presentation

Witt, Andreas (Bielefeld University) andreas.witt@uni-bielefeld.de  
L?gen, Harald (Justus-Liebig-Universit? Gie?n) harald.luengen@uni-giessen.de  
Sasaki, Felix (Bielefeld University) Felix.Sasaki@uni-bielefeld.de  
Goecke, Daniela (Bielefeld University) daniela.goecke@uni-bielefeld.de

### Contact Information recorded for this paper is:

Computational Linguistics and Text-Technology  
Bielefeld University  
Postfach 100131  
D-33501 Bielefeld

*Telephone:* ++49-521-106-3671  
*FAX:* ++49-521-106-2996

### The equipment requirements recorded for this paper are:

**Computer Requirements:**  
(nothing recorded)

**Special Equipment:**  
(nothing recorded)

## Introduction

Annotating multiple hierarchies with SGML-based markup systems is still one of the fundamental problems of text-technological research. Up to now, several solutions have been discussed (e.g. chapter 31 of the TEI-Guidelines (Sperberg-McQueen and Burnard 1994) and Barnard et al. (1995)). Furthermore, some non-SGML based approaches have been proposed. (cf. Huitfeldt and Sperberg-McQueen (2001) ; Tennison and Piez (2002)) The solution suggested here<sup>1</sup> is XML-based and best summarised by the following quotation: "In some cases, the simplest method of disentangling two conflicting hierarchical views of the same information is to encode it twice, each time capturing a single view." (Sperberg-McQueen & Burnard 1994, p.775). The main drawback of this approach, namely the missing explicit relation between the two annotation layers, is overcome by linking the separate documents by means of their common textual content (cf. Witt 2002). Based upon a common representation of two annotation layers and their textual content, a prototype has been implemented that allows for the unification of separately XML-annotated text documents. The result of the unification is one new well-formed XML document containing the text data and the annotations from multiple layers. The proposed poster will show both the theoretical and practical aspects of this technique.

## A Single Representation for Multiple Annotations

The advantage of annotating the same data multiply but separately is thatfile:///C:/home/CONFER~1/allc2004/VERSCH~1/abstract-content.html the modelling of information on one level is not dependent on (the existence of) another modelling level. This results in an information

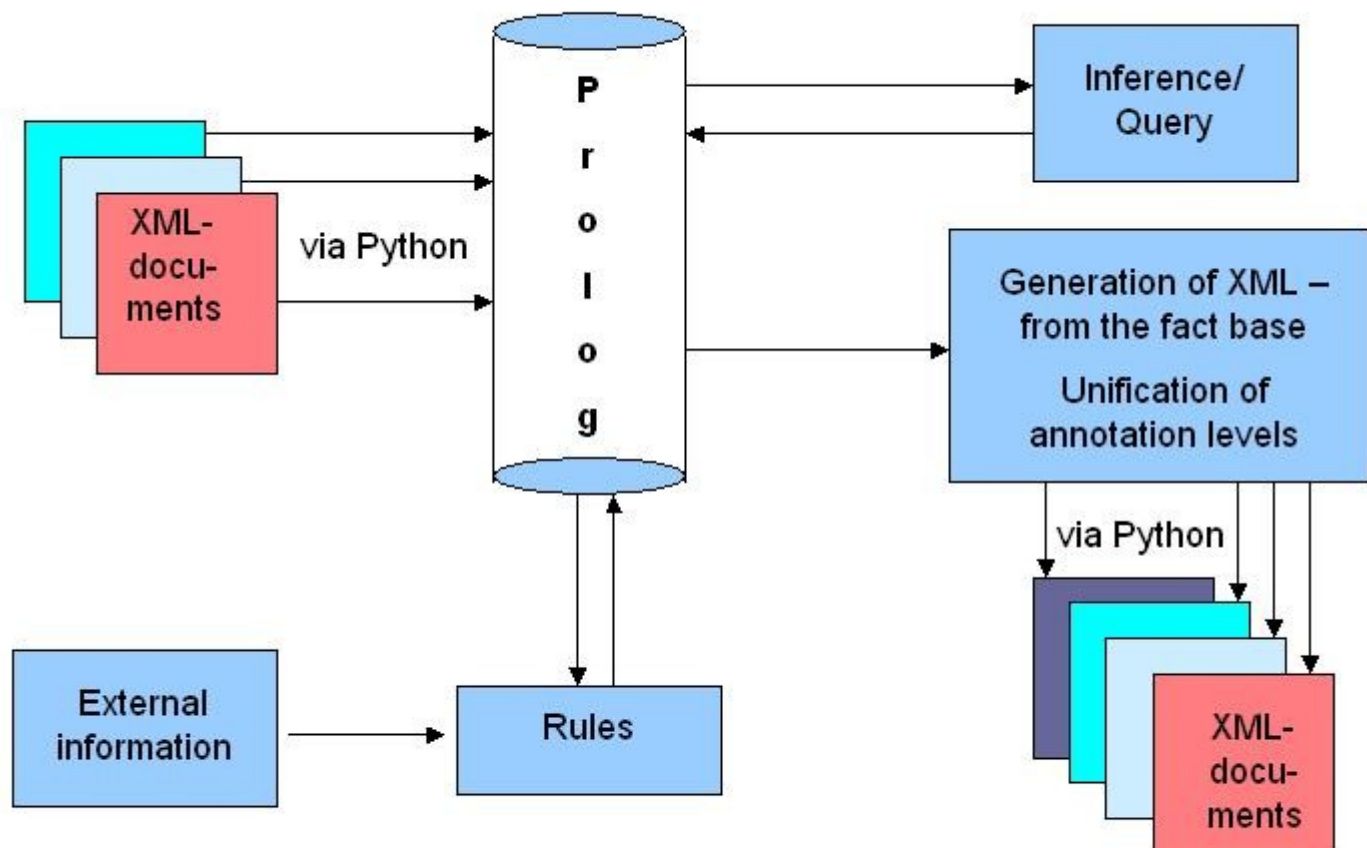
modelling where each layer can be viewed separately and new layers can be added at any time. Moreover, the creation of alternative annotations based upon different theoretical assumptions for an existing information level is possible. (cf. Sasaki et al. 2003).

The XML documents are transformed into a Prolog fact base representing a common view on the textual content and its multiple annotations. The format of this fact base, originally introduced by Sperberg-McQueen et al. (2001) for the interpretation of the semantics of (single) XML-documents, has been extended as described in Witt (2002). In this extended format, each XML-Element is represented as a Prolog predicate called `node/5`, with five arguments for referring to (1) the annotation layer, (2) the start position and (3) the end position of the annotated text span in terms of byte offsets, (4) the position of the element in the document tree, and (5) the name of the element. The transformation of the XML files to the single file containing the Prolog facts is implemented in the programming language Python. The information concerning the start and end positions is the basis for the interrelation of the documents. Special Prolog inferences have been implemented to query the relations that hold between markup elements belonging to different layers (cf. Bayerl et al. 2003).<sup>2</sup>

## Unification

### Outline

The process of unifying the different XML-annotations of a document is visualised in Figure 1. It shows all involved transformations and formats, including the merging component `semt` which in the unification process takes decisions in case of conflicts, based on automatically inferred relations between the different annotation layers, or on an externally specified rule base.



The actual merging of two document layers has also been implemented in Prolog. The predicate `semt` (sekimo merging tool) receives four arguments. The first and the second argument name the two layers which are to be unified. The third argument is a list of elements which should be deleted in the process of unification. This list specifies pairs of an element name and its annotation layer. The result of the merging are again Prolog facts, written into a new file which is specified in the fourth argument. This new database contains a copy of all layers in the input database plus the result layer. In case the unification results to a layer where the elements would not be properly nested, a second result layer (a difference list) is created.

The result, the new fact base, is (re-)converted to XML. Again Python is used for the implementation of the transformation. As an additional output of this conversion, mainly for reasons of control, a copy of each input XML-document is generated.

If no difference list exists, the result of the merging of two layers can be linearised as an XML document without special attention. In case the result fact base contains a difference list, two different linearisations can be generated. (cf. again chapter 31 of the TEI-Guidelines). The default processing uses milestone elements to mark the borders of incompatible elements. Alternatively, the technique of fragmentation of elements can be invoked. For both techniques it is possible to declare an annotation layer as 'primary', i.e. the elements in the primary layer should for example not be fragmented.

## Relations between annotations

With respect to a unification/merging of layers, the simplest configuration of two element instances on different annotation layers over the same textual data are a relation of a.) *proper nesting*, and a relation of b.) *independence*:

```
L1:  <w>friend  li      ness</w>
L2:      friend<m>li</m>ness
UL:  <w>friend<m>li</m>ness</w>
```

```
L1:  to <name>Dr. van Helsing</name> and      ask      him to
L2:  to      Dr. van Helsing      and <v>ask</v> him to
UL:  to <name>Dr. van Helsing</name> and <v>ask</v> him to
```

The annotations are unified by simply keeping every element tag in its place (cf. UL, the unification layer).

Cases of c.) *non-proper nesting*, i.e. where an element on one annotation layer either starts or ends at a position where an element on the other layer also starts or ends are a little more complicated.

```
L1:  <w>friend  ly</w>
L2:      friend<m>ly</m>
UL:  <w>friend<m>ly</m></w>
```

In these cases, the desired nesting of elements in the target layer is inferred by comparing the length of the text spans on both layers: The element spanning more characters becomes the outer element, i.e. the parent node in the result tree.

```
<m>fol      d</m><m>er</m>
<s>fol</s><s>d      er</s>
```

The most complicated case, is the relation of d.) *overlap*, because the result of a simple merging of an overlap configuration cannot be represented in a tree structure and consequently not be mapped to a single well-formed XML document in a straightforward way. Thus, in case of overlap, one of the two elements is written to the difference-list.

Finally, in case of the e.) *identity* relation (where two elements from the two layers contain an identical

text span, see the example below), a merged tree can be constructed, but the decision which of the two elements is to become the parent node (i.e. the resolution of conflicting hierarchies during the merging process) is non-trivial.<sup>3</sup>

```
reference:      <ref type="anaphoric">anata</ref>
syntax:        <np>anata</np>

text:          tada watashi no kao wo mite anata ha ikutsu desu ka to itta.
transl:        just my      - face - look you  - how old -      - said
trans2:        (He) just looked at me and said: `How old are you?'
```

## Resolution of conflicting hierarchies during the merging process

There are three alternative strategies:

1. A hierarchy is created, e.g. `<np><ref type="anaphoric">anata</ref></np>`
2. One of the elements is deleted, e.g. `<np>anata</np>`
3. One of the elements is deleted, and its attributes are integrated into the other element, e.g. `<np type="anaphoric">anata</np>`

An important source of information for deciding which strategy should be chosen is the relation of the elements, here `np` and `ref`, in general. For this task, we have implemented inferences for so-called *metarelations*. For example, the metarelation 'inclusion' holds if whenever an instance of `np` shares `pcdata` with `ref`, the relation is either a proper nesting, a non-proper nesting or identity. If this metarelation holds, the first resolution strategy must be chosen.

In addition, external rules are defined to describe for which elements a solution strategy should be applied:

- A) Rules without restrictions. The rules apply to all instances of the elements, for example to all `<np>` and `<ref>` elements.
- B) Rules with attribute restrictions. The rules apply only to elements with certain attributes and values, for example all `<ref>` elements with an attribute `type="anaphoric"`.
- C) Rules which describe configurations between elements. The rules apply only to elements which are in a specified configuration with other elements, for example all `<ref>` elements which are inside a `<syntacticDescription>` element.
- D) Rules which combine B) and C), for example # all `<ref>` elements which are inside a `<corpus>` element, which has an attribute `corpus-type="syntax has priority"`.

The rules are represented in Prolog. During the merging process they are consulted if conflicts between elements occur. Alternatively, a representation as an RDF Schema (RDFS, Brickley and Guha 03) has been developed (Sasaki, to appear). Using RDFS instead of Prolog can contribute to the formal specification of the meaning and interpretation of concurrent markup (Witt 02).

## Notes

1. The involved projects are the project A2 "Sekimo" and the project C1 "SemDoc", both part of the DFG-Forschergruppe 437 "Texttechnologische Informationsmodellierung"
2. The program is called 'Sekimo-inference tool - seit.pl'
3. The example sentence is taken from the Japanese text initiative (JTI, cf. <http://etext.lib.virginia.edu/japanese/>), from the Novel 'Kokoro' (heart), written by Natsume

Souseki.)